Naval Research Laboratory

Washington, DC 20375-5320

NRL/FR/5522--04-10,073

Simulating Sensor Networks in NS-2

IAN T. DOWNARD

Network and Communication Systems
Information Technology Division

May 31, 2004

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
31-05-2004	Final	

4. TITLE AND SUBTITLE

Simulating Sensor Networks in NS-2

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
PE 0603790N

6. AUTHOR(S)

Ian T. Downard

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
Code 5522
4555 Overlook Avenue, SW
Washington, DC 20375-5320

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/FR/5522--04-10,073

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Chief of Naval Resesarch(ONR 822)
Ballston Centre Tower One
800 North Quincy Street
Arlington, VA 22217-5660

10. SPONSOR / MONITOR'S ACRONYM(S)

ONR

11. SPONSOR / MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Optimizing sensor networks involves addressing a wide range of issues. These issues stem from limited energy reserves, computation power, communication capabilities, and self-managing sensor nodes. The NS-2 simulation environment is a flexible tool for network engineers to investigate how various protocols perform with different configurations and topologies. This report describes how we extended the NS-2 framework to include support for sensor networks.

15. SUBJECT TERMS

Simulation, Sensor/Networks

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
					Ian T. Downard
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**	UL	20	**19b. TELEPHONE NUMBER** *(include area code)*
Unclassified	Unclassified	Unclassified			(202) 767-2952

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

CONTENTS

SIMULATING SENSOR NETWORKS IN NS-2

1. INTRODUCTION

Recent advances in processing, storage, and communication technologies have advanced the capabilities of small-scale and cost-effective sensor systems to support numerous applications. Sensor networks that detect hazardous chemical or biological agents in complex urban infrastructures could be a killer application for homeland security. Much of the research in sensor networks is funded for military tasks, but applications such as forest fire detection and rush-hour traffic monitoring exemplify the versatility envisioned for this rapidly expanding technology.

Many successful sensor applications have been deployed in very specialized networks. These include the University of California-Berkeley's Smart Dust [1], the Massachusetts Institute of Technology's μ-Adaptive Multi-domain Power aware Sensors [2], and the University of California Los Angeles' Wireless Integrated Sensor Networks [3]. But the widespread deployment of wireless networks has generated more possibilities for mobile ad hoc networks of self-governing nodes that can serve numerous sensor applications without manual reconfiguration.

While operating in this context, we define a sensor network as an autonomous, multihop, wireless network with nondeterministic routes over a set of possibly heterogeneous physical layers. In other words, routing will occur throughout the network at nodes configured in ad hoc mode. Our long-term objective is to evaluate how well current routing layer standards support the requirements of various other layers in these sensor networks. We are including the NS-2 simulation environment in these evaluations.

The primary purpose of this project is to establish a foundation in NS-2 for simulating sensor networks. This foundation, illustrated in Fig. 1, consists of dual-homed sensor nodes that are tapped into an 802.11 channel for communicating with other network stations and into a phenomenon channel for detecting some physical phenomenon. This work is a small contribution that should benefit sensor network research where simulation is appropriate. It is an effort to aid the analysis of various sensor network configurations under the demands of specific sensor applications.

The report begins with an overview of the NS-2 simulation environment, followed by a description of our extensions to NS-2 and guidelines for using them in simulations. We conclude with a section to illustrate a sensor network simulation and a final section to list relevant areas for future research.

2. RELATED WORK

Reference 4 describes a project whose objective includes building a flexible simulation tool specifically for sensor networks. This ongoing research emphasizes heterogeneity throughout a simulation environment based on the SWARM [5] software package. It caters to simulations of Mobile Ad Hoc Network (MANET) nodes, each with unique storage, processing, and sensing capabilities in order to investigate details about energy conservation, routing, medium access, and application protocols.

Manuscript approved September 9, 2003.

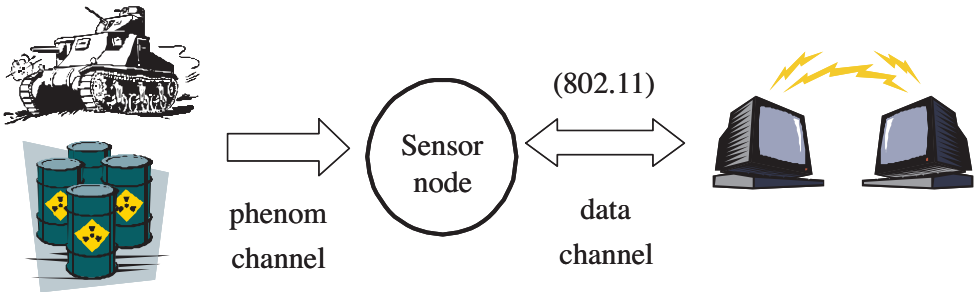

Fig. 1 — Foundation of the sensor network model used in NS-2

Reference 6 developed extensions to NS-2 for modeling sensor networks, with an emphasis on sophisticated modeling of energy consumption and emulation (i.e., interfacing with real-world sensor nodes). Unfortunately, their work has not been updated to support subsequent releases of NS-2 since October 2000.

3. NS-2 OVERVIEW

The NS-2 simulation environment [7] offers great flexibility in investigating the characteristics of sensor networks because it already contains flexible models for energy-constrained wireless ad hoc networks. In the NS-2 environment, a sensor network can be built with many of the same sets of protocols and characteristics as those available in the real world. The mobile networking environment in NS-2 includes support for each of the paradigms and protocols shown in Fig. 2. The wireless model also includes support for node movements and energy constraints. By leveraging the existing mobile networking infrastructure, we added the capability to simulate sensor networks.

4. THE EXTENDED NS-2 ARCHITECTURE

Sensor Network Extension

The only fundamental aspect of sensor networks missing in NS-2 was the notion of a phenomenon, such as chemical clouds or moving vehicles, that could trigger nearby sensors through a channel, such as air quality or ground vibrations. Once a sensor detects the "ping" of a phenomenon in that channel, the sensor acts according to the sensor application defined by the NS-2 user. This application defines how a sensor will react once it detects its target phenomenon. For example, a sensor may periodically send a report to some

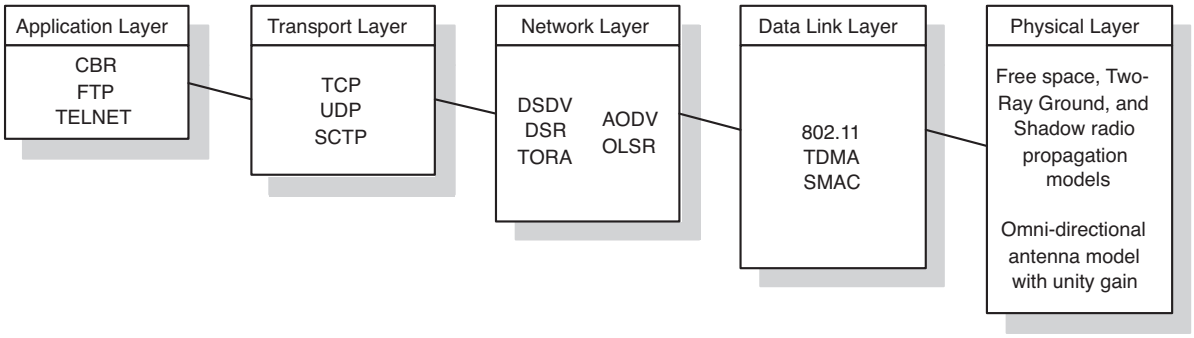

Fig. 2 — Some of the paradigms and protocols available for wireless networking in NS-2. Some protocols like OLSR [8] and SMAC [9] have not been incorporated into USC's NS-2 distributions [7], but they can be retrieved from their respective developers' sites.

data collection point as long as it continues to detect the phenomenon. Or, it may do something more sophisticated, such as collaborate with neighboring sensor nodes to more accurately characterize the phenomenon before alerting any outside observer of a supposed occurrence. For each sensor network there is a unique sensor application to accomplish phenomena detection, such as surveillance or environmental monitoring. With NS-2, we have provided the facility to invoke sensor applications by phenomena. With these sensor applications, we can study how the underlying network infrastructure performs under various constraints.

We modeled the presence of phenomena in NS-2 with broadcast packets called PHENOM, which are transmitted through a designated channel. The range of phenomena is the set of nodes that can receive the PHENOM broadcast packets in that channel.[1] This pattern will follow whichever radio propagation model (free space, two-ray ground, or shadowing) is included with the phenomenon node's configuration. These propagation models roughly cover a circle, but other shapes could be achieved by varying the range of PHENOM broadcast packets and creatively moving a set of phenomenon nodes emanating the same type of phenomenon.

Emanating PHENOM broadcast packets is accomplished by the "PHENOM routing protocol,"[2] which simply broadcasts PHENOM packets with a certain configurable pulse rate. When a PHENOM packet is received by a node listening on the phenomenon channel, a receive event is passed to that node's sensor application.

Additions to NS-2

Our sensor network simulations have phenomenon nodes that trigger sensor nodes, but the traffic that sensor nodes generate once they detect phenomena depends on the function of the sensor network. For example, sensor networks designed for energy-efficient target tracking [10] would generate more sensor-to-sensor traffic than a sensor network designed to provide an outside observer with raw sensor data. This function is defined by the sensor application, which is intended to be customized according to the traffic properties associated with the sensor network being simulated. The objects and functions we have just described are implemented in the following files:

- `phenom/phenom.cc,h`: This file implements the PHENOM routing protocol used for emanating phenomena. It includes parameters for the pulse rate and the phenomenon type (carbon monoxide, heavy seismic activity, light seismic activity, sound, or generic). These types are just names that can be used to identify multiple sources of phenomena in trace files. The pulse rate is the only parameter that actually controls how a phenomenon emanates.

- `sensornets-NRL/sensoragent.cc,h`: The ns manual [11] describes *agents* as "endpoints where network-layer packets are constructed or consumed." Sensor nodes use a *sensor agent* attached to the phenomenon channel for consuming PHENOM packets, and a User Data Protocol (UDP) or Transmission Control Protocol (TCP) agent attached to the wireless network channel for constructing packets sent down from the sensor application. Sensor agents act as a conduit through which PHENOM packets are received and processed by sensor applications. The sensor agent does

[1] This reflects the range of sensitivity of the sensors. For example, `PHENOM` broadcasts with a long range would simulate highly sensitive sensors. The sensitivity of a single sensor can be controlled by setting the receive and carrier sense thresholds defined in the `mac/wireless-phy.cc` file located in the NS-2 source tree.

[2] This functionality best fits into NS-2's existing ad hoc wireless networking infrastructure as a routing protocol, even though it does not actually route at all. The `MAC` layer it operates above must be specified in the phenomenon node's configuration. Although real-world phenomena can interfere in a variety of ways, we ignore this aspect and use the basic "Mac" class, which seems to prevent channel contention.

not actually look at the contents of the PHENOM packet; it simply marks the packet as received and passes it to the sensor application. This agent is implemented in `sensoragent.cc`.

- `sensornets-NRL/sensorapp.cc,h`: The sensor application defined in this file uses node color and generates sensor reports to show when the corresponding sensor node detects a phenomenon.[3] Specifically, when the node is receiving PHENOM packets, this application changes the node color to red, activates an "alarm" public variable, and sends a sensor report of MESG_SIZE bytes to the sink node of a UDP or TCP connection once per TRANSMIT_FREQ second. When the node has not received a PHENOM packet in the timeout period specified by SILENT_PHENOMENON, the node color changes back to green. If node color is desired to illustrate energy levels instead of sensor alarm status, that aspect of the application can be disabled with `DISABLE_COLORS`. Figure 3 is a visualization of this sensor application.

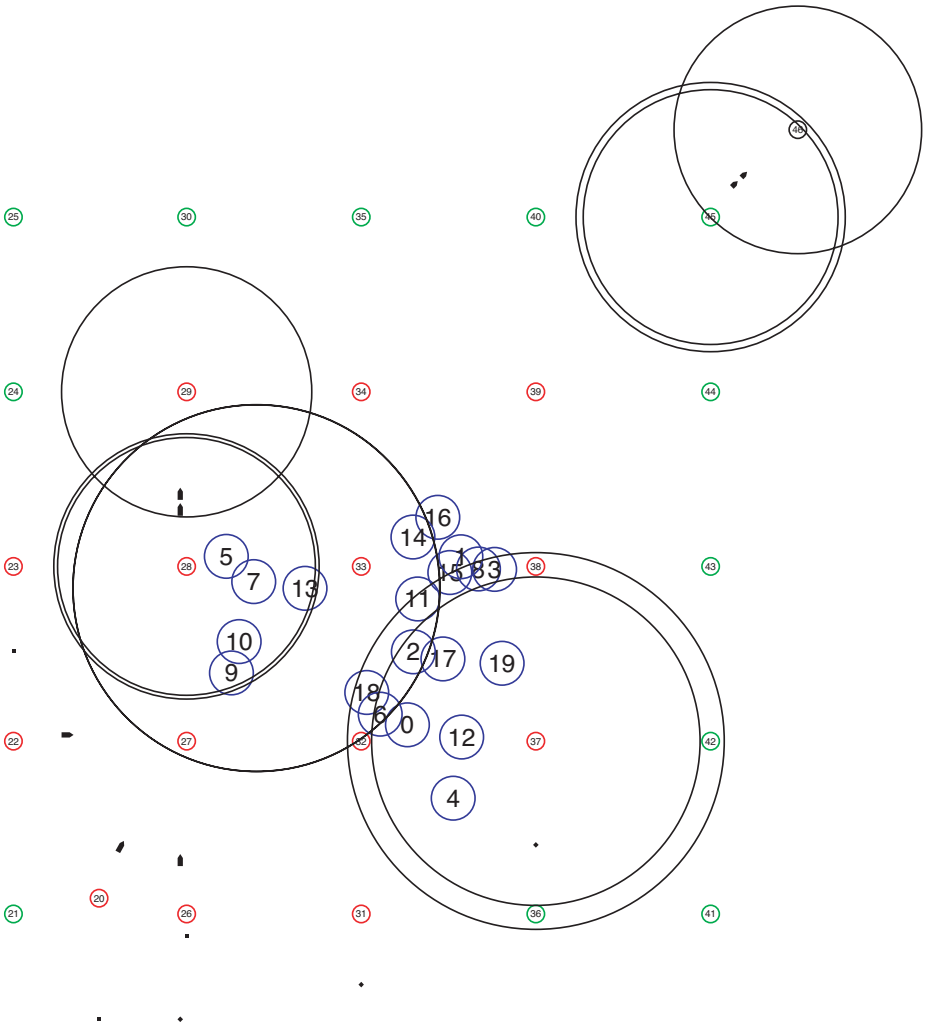

Fig. 3 — Visualization of a simulated sensor network with 25 stationary sensor nodes, 20 mobile phenom nodes simulating a gas cloud, and one stationary data collection point. The red sensor nodes detect the phenomenon, the green ones do not. The phenomenon nodes are large and blue, and the data collection point is the black node in the far upper-right corner.

[3]The four environment variables that can be used to customize this application are SILENT_PHENOMENON, DISABLE _COLORS, MESG_SIZE, and TRANSMIT_FREQ.

- `sensornets-NRL/phenom_packet.h:` This file defines the structure of PHENOM packets. The five phenomenon types defined here (CO, HEAVY_GEO, LIGHT_GEO, SOUND, and TEST_PHENOMENON) correspond to carbon monoxide, heavy seismic activity, light seismic activity, audible sound, and some generic phenomenon. These types are most useful for simulations involving multiple phenomenon nodes.

Modifications to NS-2

Figure 4 shows where our extensions are arranged within the NS-2 framework. The major additions and modifications are explained below, and the next subsection shows how our extensions fit into NS-2's class hierarchy.

- `trace/cmu-trace.cc,h`: The CMUTrace class is used to print important parts of a packet to the simulation's trace file. Since we introduced a new packet type for phenomena, we had to describe the corresponding packet format in this class.

- `tcl/lib/ns-lib.tcl`: This component of the infrastructure interprets node configurations specified in the NS-2 simulation script. Our extensions introduced two new node types, the sensor node and the phenomenon node. Therefore, we added some arguments in the node-config function to accommodate them.

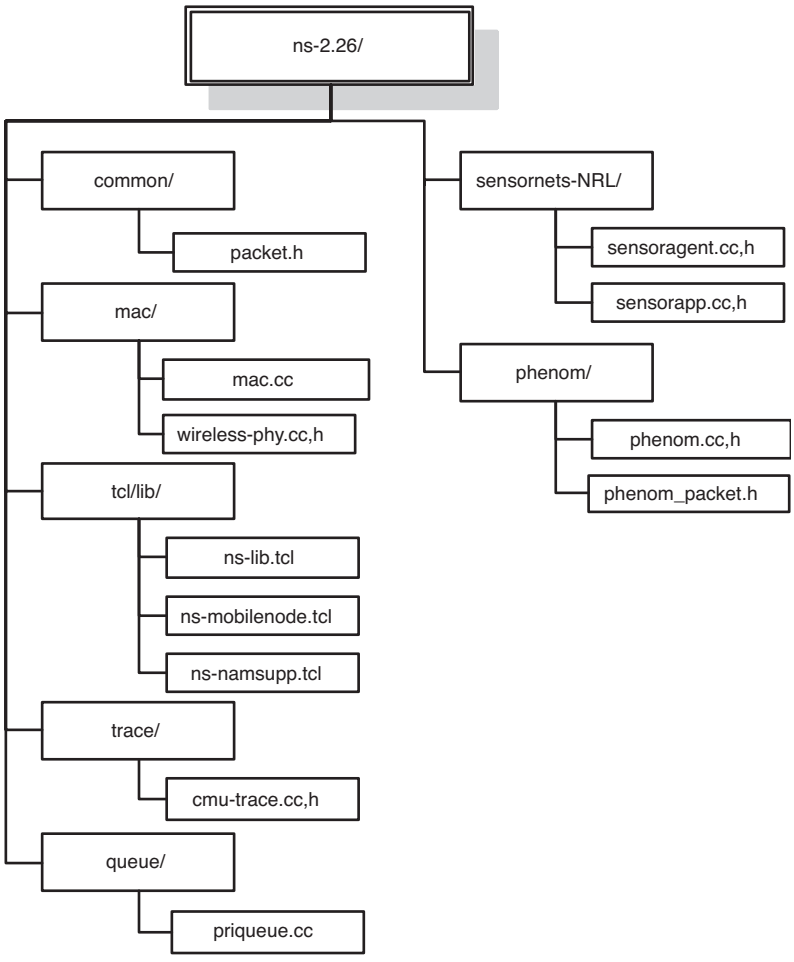

Fig. 4 — Files in the NS-2 framework that were modified (left-hand side) or added (right-hand side)

- `tcl/lib/ns-mobilenode.tcl`: In NS- 2's virtual world, we are using its existing capacity for multichannel wireless networking as a means to emanate phenomena of various kinds. By using a dedicated channel for phenomena, we can simulate the unique physical medium that they occupy in the real world. Thus, as shown in Fig. 1, sensor nodes will need to have two interfaces, one to the 802.11 channel and one to the PHENOM channel. We implemented this kind of "multi-homed" capability in `ns-mobilnode.tcl`.

- `common/packet.h`: Each packet in NS-2 is associated with a unique type that associates it with the protocol to which it belongs, such as TCP, ARP, AODV, FTP, etc. Since we created a new protocol for emanating phenomena, we defined its corresponding packet type in the `packet.h` header file.

- `mac/wireless-phy.cc`: NS-2 contains an energy model for wireless nodes that can be used to investigate the benefits of various energy conservation techniques, such as node sleeping or utilizing optimal network densities. The model includes attributes for specifying the power requirements of transmitting packets, receiving packets, or idly standing by during times of network inactivity. Sensing phenomena is a process that may consume power at another rate, so it is important to consider this where sensor network simulations are concerned. In `mac/wireless-phy.cc`, we have included the capability of specifying the amount of power consumed by nodes while sensing phenomena.

Other small modifications were made to `mac/mac.cc, tcl/lib/ns-namsupp.tcl`, and `queue/priqueue.cc` to facilitate the second interface to the phenomenon channel on sensor nodes, to fix a bug in NS-2's node coloring procedure, and to include the new PHENOM packet type into the NS-2 framework, respectively.

The Extended NS-2 Class Hierarchy

The Doxygen documentation system [12] was used to generate Figs. 5, 6, and 7. These figures illustrate how our extensions fit into NS-2's class hierarchy. Dotted lines show where a class is using the methods and members of another class. Solid lines show where a class is inheriting the methods and members from another class.

5. CAPABILITIES, GUIDELINES, AND CAVEATS

This section describes the capabilities of our sensor network extensions, gives some guidelines for configuring simulations, and attempts to explain some areas of likely confusion. We assume that the reader is familiar with setting up mobile node simulations in NS-2. For readers who are not, the following URLs provide background:

```
http://nile.wpi.edu/NS/
http://www.isi.edu/nsnam/ns/
```

The easiest way to create sensor network simulations is to use the `script_maker.pl` utility in the `simulations_aids` directory distributed with our extensions. This Perl script contains commonly used parameters for setting up sensor network simulations and automatically generates the often complex ns simulation script. The remainder of this section describes how to code a sensor network simulation into the ns simulation script without using the `script_maker.pl` utility.

Setting up a sensor network in NS-2 follows the same format as mobile node simulations. The best way to create your own simulation is to modify one of the examples distributed with our code [13].

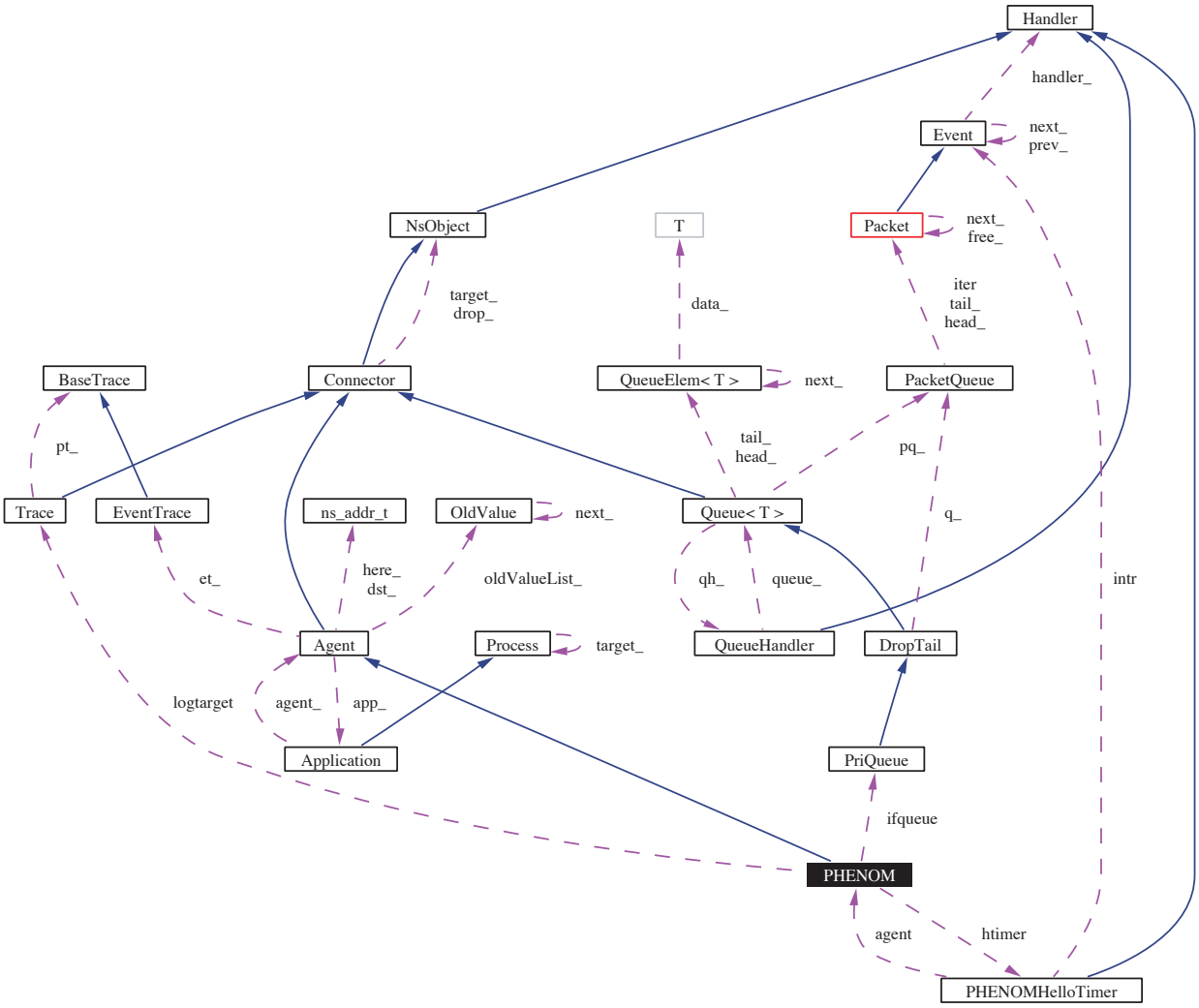

Fig. 5 — Collaboration diagram for the PHENOM class

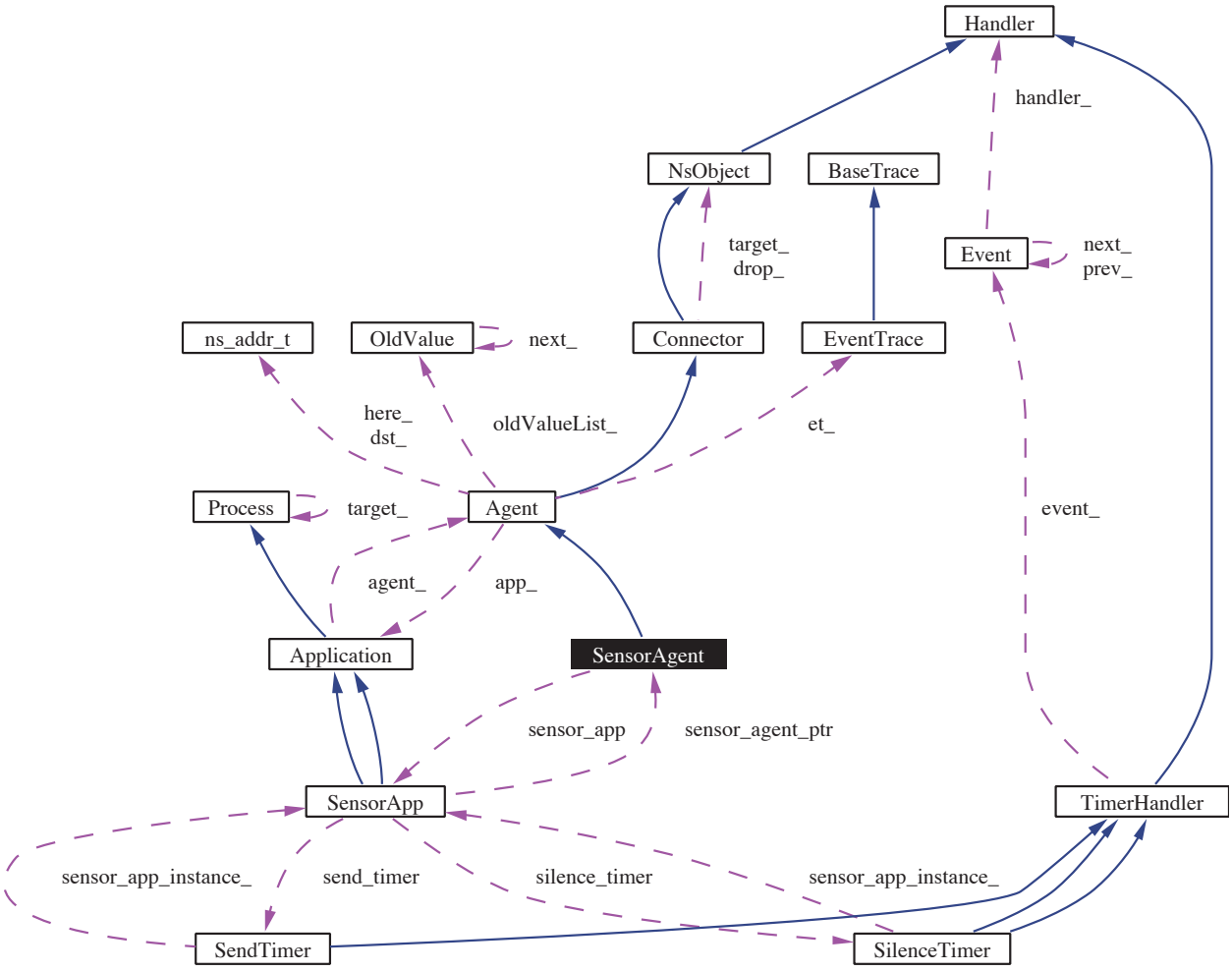

Fig. 6 — Collaboration diagram for the SensorAgent class

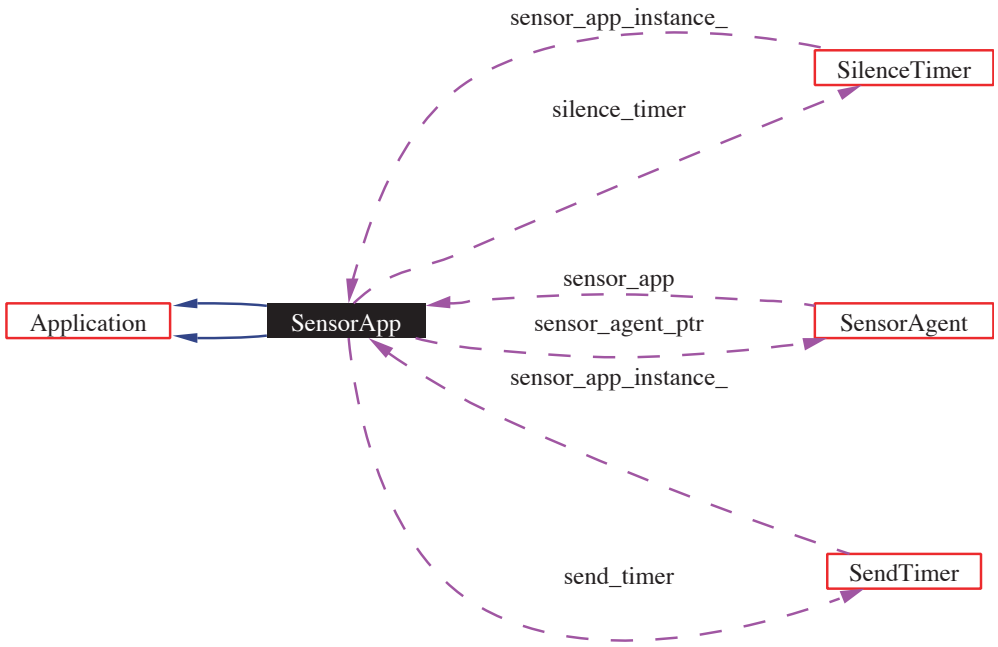

Fig. 7 — Collaboration diagram for the SensorApp class

Places where a sensor network simulation differs from a traditional mobile node simulation are listed below. Setting up `ns_`, `god_`, tracing, topography objects, and starting and stopping the simulation are all the same as in traditional mobile node simulations.

1. *Configure a phenomenon channel and data channel*. Phenomenon nodes should emanate in a different channel than sensor nodes to avoid contention at the physical layer. All phenomenon nodes should be configured on the same channel, even if they are emanating different types of phenomena.

```
set chan_1_[new $val(chan)]
set chan_2_[new $val(chan)]
```

2. *Configure a MAC protocol for the phenomenon channel*. Choose a MAC layer to use for emanating phenomena over the phenomenon channel. Using 802.11 is not appropriate, since phenomena should be emanating without regard to collisions or congestion control. We suggest using the basic "Mac" class instead, shown as follows:

```
set val(mac)       Mac/802_11
set val(PHENOMmac) Mac
```

3. *Configure phenomenon nodes with the PHENOM "routing" protocol*. Use node-config, just like with mobile nodes, but specify PHENOM as the routing protocol so the phenomenon is emanated according to the methods defined in `phenom/phenom.cc`. Also, be sure to configure in the channel and MAC layer previously specified for phenomena broadcasts. A sample node configuration statement is shown below.

```
$ns node-config \
-adhocRouting PHENOM \
-channel $chan_1_\
```

```
-llType LL \
-macType $val(PHENOMmac) \
-ifqType Queue/DropTail/PriQueue \
-ifqLen 50 \
-antType Antenna/OmniAntenna \
-phyType Phy/WirelessPhy \
-topoInstance $topo \
-agentTrace ON \
-routerTrace ON \
-macTrace ON \
-movementTrace ON \
-propType Propagation/TwoRayGround
```

4. *Configure the Phenomenon node's pulse rate and type*. The two parameters that can be used to customize phenomena are listed below. They are both optional.

   (a) `pulserate FLOAT`

   • `FLOAT` must be a real number.
   • Describes how frequently a phenomenon node broadcasts its presence.
   • Defaults to one broadcast per second.

   (b) `phenomenon PATTERN`

   • `PATTERN` must be any one of the following keywords: CO, HEAVY GEO, LIGHT GEO, SOUND, TEST_PHENOMENON, corresponding to carbon monoxide, heavy seismic activity, light seismic activity, audible sound, and some other generic phenomenon.

   • This option is mostly useful for simulations involving multiple phenomenon nodes. It makes it easier to distinguish who a sensor node is detecting by looking at the ns trace file.

   • Defaults to TEST_PHENOMENON.

The following source code illustrates how these phenomena parameters can be set to emanate carbon monoxide 10 times per second:

```
[$node_(0) set ragent_] pulserate .1 ;
[$node_(0) set ragent_] phenomenon CO ;
```

5. *Configure sensor nodes*. Sensor nodes must be configured with the `-PHENOMchannel` attribute and the `-channel` attribute. `PHENOMchannel` must be the same as the channel you configured the phenomenon node with. The other channel is the channel that will be used for communicating sensor reports. Sensor node configurations must also specify a MAC protocol for the phenomena channel and a MAC protocol (such as `Mac/802_11`) for the channel shared with other wireless nodes. This is done with the `-PHENOMmacType` and `-macType` attributes. `PHENOMmacType` should be the same as the macType used in PHENOM nodes, and `macType` should be the same as the macType used in other nodes participating in the IP network. For example:

```
$ns_node-config \
—adhocRouting $val(rp) \
```

```
–channel $chan_2_\
–macType $val(mac) \
–PHENOMchannel $chan_1_\
–PHENOMmacType $val(PHENOMmac)
```

If desired, a sensor node can be configured so that a specified amount of energy will be deducted from its energy reserve each time it receives a phenomenon broadcast. To set this up, include the following parameters in the sensor node's `node-config` routine:

```
-energyModel EnergyModel \
-rxPower 0.175 \
-txPower 0.175 \
-sensePower 0.00000175; \
-idlePower 0.0 \
-initialEnergy 0.5
```

where

- `rxPower .175` indicates 175 mW consumed for receiving a packet of arbitrary size,
- `txPower .175` indicates 175 mW consumed for transmitting a packet of arbitrary size
- `sensePower .00000175` indicates 1.75 μW consumed for receiving a PHENOM broadcast packet, and
- `initialEnergy 5` indicates a total energy reserve of 5 J.

An important caveat to keep in mind is that NS-2's energy consumption model uses color to illustrate when a node is about to exhaust its energy. To avoid confusion in the `nam` visualization, the node coloring that is part of the sensor application should be disabled with the `DISABLE_COLORS` definition in `sensorapp.cc.` (Remember to run `make` again to compile those changes into the NS-2 executable). In addition to `DISABLE_COLORS`, some other sensor node parameters can be specified in `sensorapp.cc`. These parameters are listed below:

- `SILENT_PHENOMENON` is the seconds of quiescence required for a sensor to go off its alarming state.
  Example:
  #define `SILENT_PHENOMENON .2`

- `MESG_SIZE` is the size (in bytes) of the messages to send to the gateway, or data collection point, or whatever you want to call the sink node attached to this sensor node (over UDP, for example).
  Example:
  `#define MESG_SIZE 256`

- `TRANSMIT_FREQ` is the frequency with which a sensor node triggered by PHENOM packets will send a message to the sink node. Units are in seconds, so a message of size MESG_SIZE bytes will be transmitted to the gateway node once for every TRANSMIT_FREQ seconds in which the sensor node has received one or more PHENOM packets.
  Example:
  `#define TRANSMIT_FREQ 0.1`

6. *Configure nonsensor nodes, such as data collection points, or gateways for the sensor network.* Nodes that are not sensor nodes or phenomenon nodes should not be configured with a `PHENOMchannel` since their only interface is to the MANET network. This is done with the

-PHENOMchannel "off" attribute, as follows:

```
$ns_node-config \
  – adhocRouting $val(rp) \
  – channel $chan_2_\
  – PHENOMchannel "off"
```

7. *Attach sensor agents*. Create a sensor agent for each sensor node and attach that agent to its respective node. Also, specify that all packets coming in from the PHENOM channel should be received by the sensor agent. In the following example, `$i` would represent the node number for the sensor node currently being configured.

```
set sensor_($i) [new Agent/SensorAgent]
$ns_attach-agent $node ($i) $sensor_($i)

# specify the sensor agent as the up-target for the
# sensor node's link layer configured on the PHENOM
# interface, so that the sensor agent handles the
# received PHENOM packets instead of any other agent
# attached to the node.

[$node ($i) set ll_(1)] up-target $sensor_($i)
```

8. *Attach a UDP agent and sensor application to each node (optional)*. How the sensor nodes react once they detect their target phenomenon is a behavior that should be defined in the sensor application. One such application might involve sensor nodes alerting a data collection point via UDP with information about the phenomenon. The following example illustrates how an application like that could be set up. Again, `$i` represents the node number for the sensor node currently being configured.

```
set src_($i) [new Agent/UDP]
$ns_attach-agent $node_($i) $src_($i)
$ns_connect $src_($i) $sink
set app_($i) [new Application/SensorApp]
$app_($i) attach-agent $src_($i)
```

9. *Start the sensor application*. The sensor node can receive PHENOM packets as soon as the sensor agent is attached to the node. Phenomenon nodes start emanating immediately once the simulation starts. A delayed start can be realized by reducing the range of phenomenon broadcasts to such a small area that they are effectively inaudible to any sensors (unless they occupy the exact same coordinate in the grid). A phenomenon node can be turned off this way with a command like `$ns_ at 6.0 {[$node_($i) set netif_(0)] set Pt_ 0.0001}. Pt_` is the range of the broadcast, and `$i` is the node `id` of the Phenomenon node.

   Since the sensor agent does nothing but notify the sensor application of received phenomenon broadcasts, the sensor node does not visibly react to PHENOM packets until the sensor application has been attached and started. The following example shows how to start a sensor application:

```
$ns at 5.0 "$app_($i) start $sensor_($i)"
```

6. PROOF OF CONCEPT: MANET ROUTING WITHIN A DYNAMIC SENSOR NETWORK

This experiment begins to show the types of results that can be achieved from sensor network simulations in NS-2. Suppose we would like to characterize how well AODV scales with the size of a sensor network running the sensor application defined in Section 4. We look at networks of stationary sensors with infinite energy placed in a grid with d units of distance between adjacent nodes. The network size will vary between 50 and 2000 sensor nodes. We limit the broadcast range of 802.11 radios and the range of the phenomenon to $\sqrt{2d^2}$, as shown in Fig. 8. Since we are using the Two-ray Ground radio propagation model, nodes within this boundary always receive the broadcast, and nodes outside never receive the broadcast. (In reality, this boundary is a random variable due to complex fading and interference effects).

We excite the network with a single phenomenon node that slowly travels near the perimeter of the network. As the grid density increases, the phenomenon encounters sensor nodes more frequently. Thus, as the grid density increases, AODV floods more route requests through the network. As the network becomes more congested, we should observe higher latency and higher loss rates in sensor reports delivered to the stationary data collection point. Figures 9 through 11 show latency, data rate, and loss fraction statistics.

This experiment's purpose as a proof of concept for our NS-2 extensions is complete. We have captured details of the AODV routing protocol through multiple sensor network simulations, and those results follow our expectations. A more useful result would involve classifying AODV as better or worse than some other routing protocol, but this work is left for future research. As it stands, we have demonstrated AODV performance in large networks of up to 2000 sensors excited by a mobile phenomenon. Reproducing the traffic patterns exhibited in these simulations would be extremely difficult without using similar extensions to NS-2.

7. FUTURE WORK

Much more effort should be made to improve how phenomenon emanates. It presently follows the behavior of an 802.11 broadcast, configured with one of the following radio propagation models:

- Free Space Model
- Two-Ray Ground Model
- Shadowing Model.

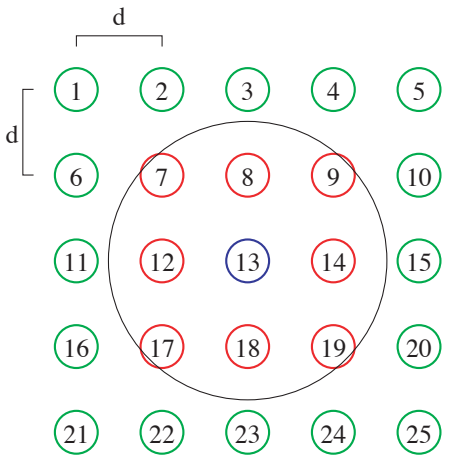

Fig. 8 — Illustration of the maximum broadcast range used in our case study. If we use the Two-ray Ground radio propagation model, node 13 can never broadcast farther than the ideal circle with radius $\sqrt{2d^2}$.

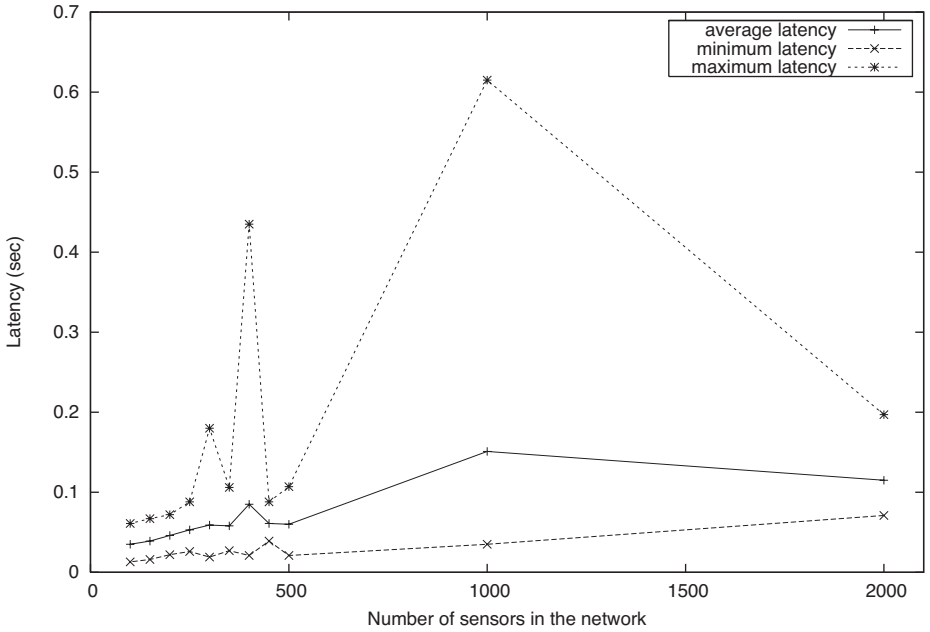

Fig. 9 — Latency as a function of network size

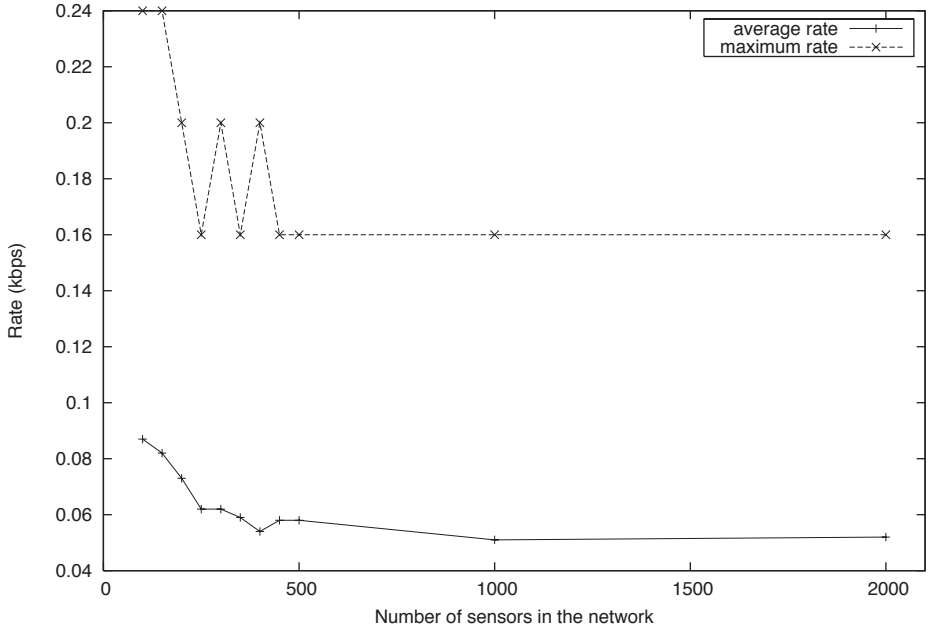

Fig. 10 — Data rates as a function of network size

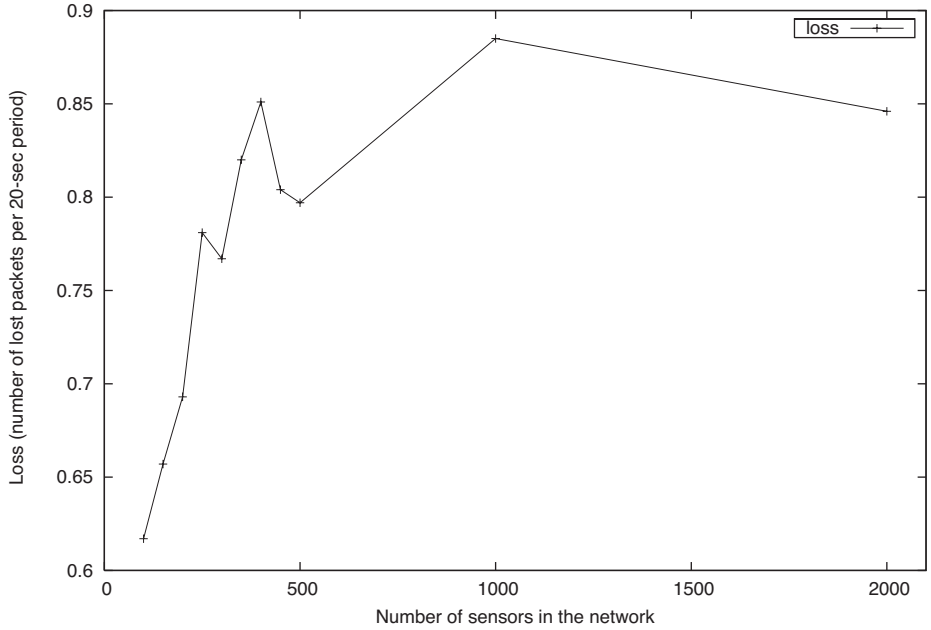

Fig. 11 — Loss fractions as a function of network size

The first two models represent the communication range as an ideal circle, whose boundary is an absolute limit on signal range. The Shadowing model applies a more probabilistic means of determining whether a receiver on the boundary can receive the signal.

Using a radio propagation model to simulate anything other than electromagnetic wave propagation is probably unrealistic. So, the radio propagation model should be extended to create various phenomenon propagation models that could specifically address the characteristics of phenomena such as seismic wave propagation or gas dispersion.

Our work to build a basic framework in NS-2 for triggering a network of sensors with phenomena can leverage more direct research in sensor networking protocols and techniques. Experiments in energy efficient routing [14] and medium access control [9] will lend themselves well to this extended NS-2 environment. Tradeoffs between power optimizations and throughput optimizations of communication protocols could be established in NS-2. Those characteristics in various sensor management schemes [15] could be similarly examined.

Directional antennas can improve the capacity of an ad hoc network [16]. It is also known that topologies can be configured to optimize energy efficient communication [17]. Investigating the power-saving benefits of dynamic topologies partially controlled by directional antennas could be complemented by NS-2's support for energy constrained mobile nodes. Results of this research could be tailored to sensor networks by exciting network traffic with mobile phenomena, as supported by our extensions to NS- 2. Reference 18 provides support for directional antennas in NS-2.

NS-2 offers great potential for mobile ad hoc sensor network research. MAC protocols, routing protocols, and applications can be customized in as much detail as their real-world counterparts. Throughput, latency, and energy levels can be gleaned from simulation trace files for measuring network performance and energy efficiency. With enough effort, anything is possible. Unfortunately, a fluent familiarity with NS-2 can be very difficult. Its flexibility goes hand-in-hand with a large and complicated architecture. Extending that architecture to create new protocols or applications can be quite difficult and time-consuming.

Learning how to use its existing capabilities is easier, but still difficult. Any effort to provide some more intuitive interface than Tcl-based scripting to NS-2's capabilities would be extremely beneficial to its users.

8. CONCLUSIONS

The primary contribution of this research is an extended capability in NS-2 to invoke network traffic consistent to the patterns expected for sensor networks. Coordinating these unique traffic patterns in NS-2 without extensions similar to ours would require significant effort for medium to large networks. Aside from generally increasing the flexibility of NS-2, this work facilitates our object of evaluating how well current MANET routing protocols support the requirements of various sensor network applications.

REFERENCES

1. J.M. Kahn, R.H. Katz, and K.S.J. Pister, "Mobile Networking for SMART DUST," in *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 99)*, Seattle, WA, August 1999.

2. μ-Adaptive Multi-domain Power aware Sensors at MIT. http://www-mtl.mit.edu/research/icsystems/uamps/

3. Wireless Integrated Sensor Networks at UCLA. http://www.janet.ucla.edu/WINS/

4. B.C. Mochocki and G.R. Madey, "H-MAS: A Heterogeneous, Mobile, Ad-hoc Sensor-Network Simulation Environment," in *Seventh Annual SWARM Users/Researchers Conference*, Notre Dame, Indiana, April 2003.

5. The SWARM Development Group. http://www.swarm.org

6. S. Park, A. Savvides, and M.B. Srivastava, "SensorSim: A Simulation Framework for Sensor Networks." http://nesl.ee.ucla.edu/projects/sensorsim/

7. The Network Simulator -ns-2. http://www.isi.edu/nsnam/ns/

8. NRL's OLSR implementation for ns-2. http://pf.itd.nrl.navy.mil/projects/olsr/

9. W. Ye, J. Heidemann, and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *Proceedings of the IEEE INFOCOM*, 2002.

10. H. Yang and B. Sikdar, "A Protocol for Tracking Mobile Targets using Sensor Networks," in *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications*, Anchorage, Alaska, May 2003, pp. 71-81.

11. The NS Manual. http://www.isi.edu/nsnam/ns-nsdocumentation.html

12. The Doxygen documentation system. http://www.doxygen.org

13. NRL's Sensor Network Extension to ns-2. http://nrlsensorsim.pf.itd.nrl.navy.mil/

14. A. Safwat, H. Hassanein, and H. Mouftah,"Energy-Aware Routing in MANETs: Analysis and Enhancements," in *Proceedings of the Fifth ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems,* in conjunction with *ACM MobiCom 2002*, Atlanta, Geor-

gia, September 2002.

15. D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next Century Challenges: Scalable Coordination in Sensor Networks," in *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, Seattle, Washington, August 1999, pp. 263-270.

16. S. Roy, D. Saha, S. Bandyopadhyay, T. Ueda, and S. Tanaka, "A Network-Aware MAC and Routing Protocol for Effective Load Balancing in Ad Hoc Wireless Networks with Directional Antenna," in *Proceedings of the Fourth ACM International Symposium on Mobile Ad Hoc Networking and Computing,* Annapolis, Maryland, June 2003, pp. 88-97.

17. A. Salhieh, J. Weinmann, M. Kochhal, and L. Schwiebert, "Power Efficient Topologies for Wireless Sensor Networks," in *Proceedings of the International Conference on Parallel Processing*, Valencia, Spain, September 2001, pp. 156-163.

18. Y.B. Ko, V. Shankarkumar, and N.H. Vaidya, "Medium Access Control Protocols Using Directional Antennas in Ad Hoc Networks," in *Proceedings of the IEEE INFOCOM 2000,* Tel Aviv, Israel, March 2000, Vol. 1, pp. 13-21.